## SQL NULL

## **What is a NULL Value?**

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

**Note:** A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

## **How to Test for NULL Values?**

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

### **IS NULL Syntax**

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

### **IS NOT NULL Syntax**

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NOT NULL;

## **The IS NULL Operator**

The IS NULL operator is used to test for empty values (NULL values).

Always use IS NULL to look for NULL values.

The following SQL lists all customers with a NULL value in the "Address" field:

* SELECT CustomerName, ContactName, Address  
  FROM Customers  
  WHERE Address IS NULL;

## **The IS NOT NULL Operator**

The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

The following SQL lists all customers with a value in the "Address" field:

* SELECT CustomerName, ContactName, Address  
  FROM Customers  
  WHERE Address IS NOT NULL;

## **The SQL UPDATE Statement**

# *The UPDATE statement is used to modify the existing records in a table*.

### **UPDATE Syntax**

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

**Note:** Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

## **UPDATE Table**

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person and a new city.

### **Example**

* UPDATE Customer  
  SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
  WHERE CustomerID = 1;

## **UPDATE Multiple Records**

It is the WHERE clause that determines how many records will be updated.

The following SQL statement will update the ContactName to "Juan" for all records where country is "Mexico":

### **Example**

* UPDATE Customer  
  SET ContactName='Juan'  
  WHERE Country='Mexico';

## **Update Warning!**

Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

* UPDATE Customers  
  SET ContactName='Juan';

## **The SQL DELETE Statement**

The DELETE statement is used to delete existing records in a table.

### **DELETE Syntax**

DELETE FROM table\_name WHERE condition;

**Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

## **SQL DELETE Example**

The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

### **Example**

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

## **Delete All Records**

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM table\_name;

The following SQL statement deletes all rows in the "Customer" table, without deleting the table:

### **Example**

DELETE FROM Customer;

## **The SQL SELECT TOP Clause**

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

**Note:** Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses FETCH FIRST n ROWS ONLY and ROWNUM.

## **SQL TOP, LIMIT and FETCH FIRST Examples**

The following SQL statement selects the first three records from the "Customers" table (for SQL Server/MS Access):

### **Example**

SELECT TOP 3 \* FROM Customer;

The following SQL statement shows the equivalent example for MySQL:

### **Example**

SELECT \* FROM Customer  
LIMIT 3;

The following SQL statement shows the equivalent example for Oracle:

### **Example**

SELECT \* FROM Customer  
FETCH FIRST 3 ROWS ONLY;

The following SQL statement shows the equivalent example for Oracle:

### **Example**

SELECT \* FROM Customer  
FETCH FIRST 3 ROWS ONLY;

## **The SQL MIN() and MAX() Functions**

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

## **MIN() Example**

The following SQL statement finds the price of the cheapest product:

### **Example**

SELECT MIN(Price) AS SmallestPrice  
FROM Products;

## **MAX() Example**

The following SQL statement finds the price of the most expensive product:

### **Example**

SELECT MAX(Price) AS LargestPrice  
FROM Products;

Use the correct function to return the number of records that have the Price value set to 18.

Select count \* from product

Where price=18;

Use an SQL function to calculate the average price of all products.

Select AVG (price)

From Product;

Use an SQL function to calculate the sum of all the Price column values in the Products table.

Select SUM(price)

From Product;

## **The SQL LIKE Operator**

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

* The percent sign (%) represents zero, one, or multiple characters
* The underscore sign (\_) represents one, single character

**Note:** MS Access uses an asterisk (\*) instead of the percent sign (%), and a question mark (?) instead of the underscore (\_)

The percent sign and the underscore can also be used in combinations!

### **LIKE Syntax**

|  |  |
| --- | --- |
| **LIKE Operator** | **Description** |
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

* SELECT column1, column2, ...  
  FROM table\_name  
  WHERE columnN LIKE pattern;

**Tip:** You can also combine any number of conditions using AND or OR operators.

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

## **SQL LIKE Examples**

The following SQL statement selects all customers with a CustomerName starting with "a":

### **Example**

* SELECT \* FROM Customer  
  WHERE CustomerName LIKE 'a%';

The following SQL statement selects all customers with a CustomerName ending with "a":

### **Example**

* SELECT \* FROM Customer  
  WHERE CustomerName LIKE '%a';

The following SQL statement selects all customers with a CustomerName that have "or" in any position:

### **Example**

* SELECT \* FROM Customer  
  WHERE CustomerName LIKE '%or%';

The following SQL statement selects all customers with a CustomerName that have "r" in the second position:

### **Example**

* SELECT \* FROM Customer  
  WHERE CustomerName LIKE '\_r%';

The following SQL statement selects all customers with a CustomerName that starts with "a" and are at least 3 characters in length:

### **Example**

* SELECT \* FROM Customer  
  WHERE CustomerName LIKE 'a\_\_%';

The following SQL statement selects all customers with a ContactName that starts with "a" and ends with "o":

### **Example**

* SELECT \* FROM Customer  
  WHERE ContactName LIKE 'a%o';

The following SQL statement selects all customers with a CustomerName that does NOT start with "a":

### **Example**

* SELECT \* FROM Customers  
  WHERE CustomerName NOT LIKE 'a%';

## **SQL Wildcard Characters**

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the [LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| \* | Represents zero or more characters | bl\* finds bl, black, blue, and blob |
| ? | Represents a single character | h?t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ! | Represents any character not in the brackets | h[!oa]t finds hit, but not hot and hat |
| - | Represents any single character within the specified range | c[a-b]t finds cat and cbt |
| # | Represents any single numeric character | 2#5 finds 205, 215, 225, 235, 245, 255, 265, 275, 285, and 295 |

### **Wildcard Characters in SQL Server**

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| % | Represents zero or more characters | bl% finds bl, black, blue, and blob |
| \_ | Represents a single character | h\_t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ^ | Represents any character not in the brackets | h[^oa]t finds hit, but not hot and hat |
| - | Represents any single character within the specified range | c[a-b]t finds cat and cbt |

All the wildcards can also be used in combinations!

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |  |
| --- | --- |
| **LIKE Operator** | **Description** |
| WHERE CustomerName LIKE 'a%' | Finds any values that starts with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that ends with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that starts with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that starts with "a" and ends with "o" |

## **Using the % Wildcard**

The following SQL statement selects all customers with a City starting with "ber":

### **Example**

* SELECT \* FROM Customers  
  WHERE City LIKE 'ber%';